UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/997,990	11/30/2001	Jeremy Alan Arnold	IBM / 193	4258

7590 12/30/2005

Scott A. Stinebruner
Wood, Herron & Evans, L.L.P.
2700 Carew Tower
441 Vine St.
Cincinnati, OH 45202-2917

EXAMINER	
PHAM, CHRYSTINE	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 12/30/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/997,990
Filing Date: November 30, 2001
Appellant(s): ARNOLD ET AL.

Scott A. Stinebruner
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed on October 4, 2005 appealing from the Office action mailed on May 4, 2005.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellants' statement of the status of amendments after final rejection contained in the brief is correct.

The amendment after final rejection filed on August 4, 2005 has been entered.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is substantially correct. The changes (indicated in **bold** text) are as follows:

(a) Claims 14-15, 17-22, 34-35, and 40 are rejected under 35 U.S.C **103(a)** (was 102(b)) as being unpatentable over Lenkov et al. (US 5560009) in view of Phillips et al. (US 5321828).

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

The following is a listing of the evidence (e.g., patents, publications, Official Notice, and admitted prior art) relied upon in the rejection of claims under appeal.

U.S. Patent No. 5,560,009	LENKOV ET AL.	Sep. 24, 1996
U.S. Patent No. 5,21,828	PHILLIPS ET AL.	Jun. 14, 1994
U.S. Patent No. 5,754,839	PARDO ET AL.	May 19, 1998

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 14-15, 17-22, 34-35 and 40 stand rejected under 35 U.S.C 103(a) as being unpatentable over U.S. Patent No. 5,560,009 to Lenkov et al. (hereinafter, Lenkov) in view of U.S. Patent No. 5,21,828 to Phillips et al. (hereinafter, Phillips).

Claims 16 and 36-37 stand rejected under 35 U.S.C 103(a) as being unpatentable over Lenkov in view of Phillips further in view of U.S. Patent No. 5,754,839 to Pardo et al. (hereinafter, Pardo).

These rejections are set forth in the final Office Action mailed on May 4, 2005 and will be reproduced in part for the appealed claims. See section **(12)** below.

(10) Response to Argument

1. **Independent claims 14, 34, and 40** (Brief, pages 5-9)

Appellants contends, "the combined teachings of Lenkov and Phillips simply fail to disclose or suggest that a number representative of object creations from multiple creators for a class be tracked, or that a condition for halting execution of a program can be tested against such a tracked number" (Brief, page 8, last paragragh).

- Essentially, the Appellants point to the "instance breakpoint" in col.29:16-30 and the "class breakpoint" in col.29:37-44 of Lenkov to argue that Lenkov does not teach tracking a number of object creations (conventionally known as *object instantiations*) resulting from multiple creators (conventionally called *constructor methods*).
  - However, in the final Office Action, Lenkov is relied upon for teaching *tracking object creations resulting from multiple creators*, in other words,

Lenkov teaches tracking calls or invocations to constructor methods (for object instantiations). This tracking is done by inserting a breakpoint in each of the constructor methods (i.e., creators) so that execution is suspended whenever any of the methods is invoked (i.e., for object creation) (see *class breakpoints* col.29:35-45). Furthermore, Lenkov also teaches that the debugger 320 handles operations on any set of overloaded functions as a group and that the user may set **a** breakpoint on **all overloaded functions** with a particular name (see col.29:55-67). It should be noted that overloaded functions are understood in the art as multiple functions sharing the same name. The advantage of using overloaded functions in software programming is that the programmer does not have to provide different names to different functions, which essentially perform the same task. For example, a program may define two print() functions with different input parameters such as print(int i) and print(String s). Upon encountering a call to a print() function, the execution engine would examine the type input parameter accompanying the call, i.e., whether it is an integer or a string, and direct the program flow to either the function print(int i) or function print(String s) respectively. Thus, the programmer is alleviated of the need to recall the, otherwise, different names to different print() functions (e.g., print_integer(int i) and print_string(String s)) in order to pass in different types of input parameters to print. With that said, a plurality of constructor methods is

inherently a set of overloaded functions because they all share the same [class] name and each of the multiple constructor methods takes a different type of input parameter. Thus, referring back to Lenkov, setting **a** breakpoint on **all** overloaded functions with a particular name clearly anticipates setting **a** breakpoint on **all** constructor methods (i.e., creators), that is to say, tracking object instantiations (i.e., object creations) resulting from multiple constructor methods (i.e., creators) because as stated above, it does not matter which of the different constructor methods (i.e., creators) is invoked, execution is still suspended because a breakpoint in an invoked constructor method has been encountered (i.e., hit). Note, that since the debugger only performs the breakpoint setting on the overloaded functions (i.e., constructor methods), execution is unaffected (i.e., not halted) when the other member functions (i.e., non-constructor methods) are called, simply because no breakpoint has been set in them.

- The Appellants further contend, "the reference is entirely silent with respect to the concept of tying a condition to a class breakpoint, much less a condition that is based upon the total number of hits to different methods in a class" (Brief, page 7, 2nd paragraph).
  - It is respectfully submitted that the Lenkov reference was never relied upon for teaching a conditional breakpoint (e.g., halting execution when the number of hits to a breakpoint meets a condition, that is to say when

the number of hits to a breakpoint equals a specified number N). Thus, Appellants' analysis of the references is considered piecemeal because the rejection of the claims is based on the combination of Lenkov and Phillips. Although Lenkov anticipates tracking object creations resulting from multiple creators as discussed above, Lenkov does not expressly disclose recording (i.e., keeping track of) a specific number of object creations resulting from multiple creators, that is to say, Lenkov does not suggest tracking the specific number of times that the multiple breakpoints (in the constructor methods) are encountered/hit. However, as the Appellants also admitted (see Appellants' VII. ARGUMENTS, page 8, 2nd paragraph), Phillips discloses the concept of setting a condition on a breakpoint (set in a function/method) that triggers the breakpoint only after N hits, that is to say, Phillips teaches halting execution of the program in response to the number of hits [to a breakpoint] (i.e., a counter) meeting a condition (i.e., equals N hits). Thus, it is inherent in Phillips that the **number of hits** to each of the multiple breakpoints is **tracked**, otherwise, it is impossible to determine when the number of hits (i.e., a counter) actually meets the condition. Since, Lenkov teaches tracking object creations resulting from multiple creators by means of a breakpoint and halting execution whenever any one of the constructor methods is encountered (i.e., halting execution every time **an object is created**, regardless of which creator is invoked), and Phillips teaches halting

execution only when the number of hits (i.e., a counter) to **a** breakpoint meets a condition (i.e., equals N hits), it would have been obvious to one of ordinary skill in the pertinent art at the time the invention was made to incorporate the teaching of Phillips into that of Lenkov for the inclusion of tracking the number of hits to **a** breakpoint set on multiple constructor methods, thus enabling the halting of execution only after a certain number of calls to constructor methods has been made (i.e., the number of object creations [resulting from multiple creators] meeting a condition). And the motivation for doing so, as has been established in final Office Action, would have been to facilitate the monitoring of data (e.g., program variables, conditions, states) at specific execution points (i.e., each time a breakpoint is hit or encountered) and storing these information for later retrieval, thus enabling the user to specify a special point of interest (e.g., when a specific breakpoint has been encountered n number of times, that is to say, when the instruction, or function in which the breakpoint was set has been executed or called n number of times) at which he would like to analyze program conditions (see *Phillips et al.* col.26:38-65; col.27:40-col.28:40).

- Appellants further contend, "Phillips falls short of disclosing or suggesting the concept of tracking the number of object creations for a class that result from multiple creators" (Brief, page 8, 2nd paragraph).

- It should be noted that the Phillips reference was never relied upon for teaching the concept of tracking the number of object creations for a class that result from multiple creators. Thus, Appellants' argument is considered piecemeal because the rejection of the claims is based on the combination of Lenkov and Phillips as established in the final Office Action as well as the discussion from above.

2. **Dependent Claim 17** (Brief, pages 9-10)

- Appellants essentially argue that Lenkov does not discriminate between creator and non-creator functions (Brief, page 9, 3rd full paragraph). The Appellants further assert, "just because Lenkov identifies all member functions, Lenkov does not specifically identify member functions that are creators, just as it does not identify member functions that are not creators" (Brief, page 10, 1st paragraph).
  - As has been established in the final Office Action as well as the discussion relative to independent claims 14, 34, and 40 above, col.29:55-67 of Lenkov discloses a debugger 320 handling operations (i.e., setting a breakpoint) on any set of overloaded functions as a group. Since constructor methods, or "creators", are inherently overloaded functions. It is clear from the passage that a group of constructor methods is inherently identified and distinguished against the other non-constructor methods. Furthermore, the same passage discussed using information, concerning the overloaded functions, found in the procedure descriptor table, and in col.27:5-12, Lenkov discloses the procedure descriptor

table containing a description of all functions/methods of the class indicating whether the each of the functions is constructor, destructor, and/or operator function. Thus, contrary to Appellants' argument, Lenkov clearly anticipates identifying all creators for a class.

3. **Dependent Claim 19** (Brief, pages 10-11)

- The Appellants argue that Lenkov does not teach receiving user input to select a "subset" of identified creators such that breakpoints are only set on the subset of identified creators (Brief, page 10, last paragraph).
  - It is submitted that, as is well understood in computer science, every set Y is also a subset of itself, namely "Y". Thus, a subset of Y that is not equal to Y should be called **proper** (or **strict**) subset and not just a "subset" per se. For example, the set {1,2,3} is a subset of {1,2,3}, while the set {1,2} is a **proper** subset of {1,2,3}. Thus, since Lenkov teaches receiving user input to set breakpoint on all displayed (i.e., identified) constructor methods (col.29:55-67), Lenkov anticipates identifying subset of identified constructor methods (i.e., creators), as claimed.

4. **Dependent Claim 16** (Brief, pages 11-12)

- The Appellants, referring to Pardo, argue that "there is absolutely no suggestion in the reference of the desirability of using a counter to track multiple breakpoints associated with a particular class, much less to track

multiple breakpoints associated with multiple creators" (Brief, page 12, 1st full paragraph).

- It is submitted that, similar to Appellants' previous arguments against the combination of references, this particular argument is obviously a piecemeal analysis of the Pardo reference when the rejection of claim 16 is based on a combination of references Lenkov, Phillips, and Pardo. In the final Office Action, Pardo is only relied upon for teaching the feature of **incrementing** (i.e., updating) a breakpoint/watchpoint counter every time the breakpoint/watchpoint is encountered/hit, while, Lenkov, as discussed above, teaches tracking object creations resulting from multiple creators by means of setting a breakpoint in each of the creators, and more importantly, Phillips teaches tracking a number of hits (i.e., counter) to a breakpoint. Thus, Phillips and Lenkov together teach updating a breakpoint counter in response to hitting any of the breakpoints set on the multiple creators. However, Phillips and Lenkov fall short of teaching incrementing the counter. It would have been obvious to one of ordinary skill in the pertinent art at the time the invention was made to incorporate the teaching of Pardo into that of Phillips and Lenkov for the inclusion of incrementing the counter. And the motivation for doing so would have been, as established in final Office Action, to speculatively execute the creator methods (i.e., instructions in which breakpoints are set) and flush the execution results in case of an interrupt (such as canceling an

instruction associated with the breakpoint [inside the creator method] before the instruction is completed) without generating false breakpoints (i.e., a breakpoint hit that gets counted when the associated instruction has not actually been executed, due to the instruction being canceled) (see Pardo col.1:15-60; col.2:25-62).

5. **Dependent Claims 36 and 37** (Brief, pages 12-13)

- Appellants similarly contend, "[A]s discussed above in connection with claim 16, none of the cited references discloses or suggests the use of a counter to track the hitting of breakpoints set on multiple creators for a class" (Brief, page 12, last paragraph).
  - Claims 36 and 37 recite the same limitation recited in claim 16, i.e., "incrementing a counter in response to hitting any of a plurality of breakpoints set on a plurality of creators for the class", which has been addressed above. Thus, Appellants are referred to the above response for Claim 16.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

(12) The claim rejections set forth in the final Office Action mailed on May 4, 2005 are reproduced (in part for the appealed claims) here for completeness:

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 9-11, 14-15, 17-22, 30-32, 34-35, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Lenkov et al.* in view of Phillips et al. (*Phillips et al.*, US 5321828).

Claim 9

The rejection of base claim 1 is incorporated. *Lenkov et al.* do not expressly disclose tracking a total number of hits to the plurality of breakpoints. However, *Phillips et al.* teach a method of debugging a computer program comprising tracking a number of hits to the plurality of breakpoints (see at least Abstract; *software breakpoints, program, function* col.26:40-col.27:25; *disabling breakpoints, break conditions, ignore* col.28:1-col.29:25). *Lenkov et al.* and *Phillips et al.* are analogous art because they are both directed to debugging computer program. It would have been obvious to one of ordinary skill in the pertinent art at the time the invention was made to incorporate the teaching of *Phillips et al.* into that of *Lenkov et al.* for the inclusion of tracking the number of hits to the breakpoints. And the motivation for doing so would have been to facilitate the monitoring of data (e.g., program variables, conditions, states) at specific execution points (i.e., each time a breakpoint is hit or encountered) and storing these information for later retrieval, thus enabling the user to specify a special point of interest (e.g., when a specific breakpoint has been encountered n number of times, that is to say, when the instruction, or function in which the breakpoint was set has been executed or called n number of times) at which he would like to analyze program conditions (see *Phillips et al.* col.26:38-65; col.27:40-col.28:40)

Claim 10

The rejection of base claim 9 is incorporated. *Phillips et al.* further teach halting execution of the computer program during debugging in response to hitting any of the plurality of breakpoints includes:
determining whether the total number of hits meets a condition in response to hitting any of the plurality of breakpoints (see at least *disabling breakpoints, ignore, n number of times* col.28:1-col.29:15) and
halting execution of the object-oriented computer program if the total number of hits meets the condition (i.e., threshold) (see at least *target system 14 program, breakpoint, interrupt tracing, hit, <count> number of times* col.28:65-col.29:20). It would have been obvious to one of ordinary skill in the pertinent art at the time the invention was made to incorporate the teaching of *Phillips et al.* into that of *Lenkov et al.* for the inclusion of determining the number of hits meeting a condition and halting execution of the program when the number meets the condition. And the motivation for doing so would have been the same, as has been cited in claim 9.

Claim 14
Lenkov et al. teach a computer-implemented method (program product) of debugging an object-oriented computer program (see claim 1), the method comprising tracking object creations (resulting from multiple creators) of a class defined in the object-oriented computer program (see claim 1). *Lenkov et al.* do not expressly disclose tracking a number of said object creations and halting execution of the object-oriented computer program in response to the number of object creations meeting a condition. However, *Phillips et al.* disclose tracking a number of hits for a breakpoint (see claim 9) and halting execution of the computer program in response to the number of hits meeting a condition (see claim 10). It would have been obvious to one of ordinary skill in the pertinent art at the time the invention was made to incorporate the teaching of *Phillips et al.* into that of *Lenkov et al.* for the inclusion of tracking a number of object creations and halting execution in response to the number meeting a condition. And the motivation for doing so would have been the same as has been cited in claim 9.

Claims 11, 15
Claims recite limitations which have been addressed in claim 10, therefore, are rejected for the same reasons as cited in claim 10.

Claims 17-22
Claims recite limitations which have been addressed in claims 1-4, and 12, therefore, are rejected for the same reasons as cited in claims 1-4, and 12.

Claims 30-32, 34-35, and 40
Claims recite limitations which have been addressed in claims 1, 5, 9-11, 14, 23, and 38, therefore, are rejected for the same reasons as cited in claims 1, 5, 9-10, 14, 23, and 38.

3. Claims 16, 36, and 37 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Lenkov et al.* in view of *Phillips et al.* further in view of Pardo et al. (*Pardo et al.*, US 5754839).

Claim 16

The rejection of base claim 14 is incorporated. *Lenkov et al.* and *Phillips et al.* do not expressly disclose incrementing a counter in response to hitting any of a plurality of breakpoints. However, *Pardo et al.* disclose incrementing a counter (see at least *counter module 40, counter 41, counter 42* FIG.2 & associated text) in response to hitting a watchpoint (see at least *breakpoint* col.5:37-50; *breakpoints, counters 41, 42, watchpoints* col.6:15-60). *Lenkov et al.*, *Phillips et al.* and *Pardo et al.* are analogous art because they are directed to debugging computer program using breakpoints. It would have been obvious to one of ordinary skill in the pertinent art at the time the invention was made to incorporate the teaching of *Pardo et al.* into that of *Lenkov et al.* and *Phillips et al.* for the inclusion of incrementing a counter in response to hitting a breakpoint. And the motivation for doing so would have been to improve the debugging program and the information generated thereby. That is to say, keeping a breakpoint counter and incrementing the counter in response to hitting the breakpoint enables the instructions where breakpoints are set to be speculatively executed and their results to be flushed in case of an interrupt without generating false breakpoints (see *Pardo et al.* col.1:15-60; col.2:25-62).

Claims 36-37

Claims recite limitations, which have been addressed in claims 1, and 16, therefore, are rejected for the same reasons as cited in claims 1 and 16.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Chrystine Pham
Examiner
Art Unit 2192

TUAN DAM
SUPERVISORY PATENT EXAMINER

Conferees:

Tuan Dam, SPE 2192

Kakali Chaki, SPE 2193

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER